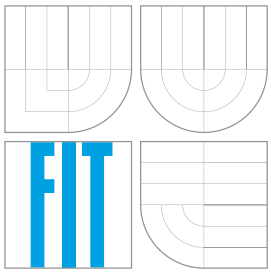
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VYHLEDÁVÁNÍ OPAKOVANÝCH ZÁBĚRŮ

COPY DETECTION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN KLETZANDER

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MICHAL HRADIŠ

BRNO 2010

Abstrakt

Tato práce popisuje metody vyhledávání podle obsahu. Vybranou metodu dále aplikuje na vyhledávání podobných obrázků a snímků ve videu. Práce dále kvalifikuje metody používané na zrychlení tohoto typu vyhledávání. Metody dále vyhodnocuje, diskutuje výsledky a zároveň navrhuje jejich používaná vylepšení.

Abstract

This thesis describes content-search methods. Chosen method is applied on finding similar images and video frames. Thesis also qualifies methods used for acceleration of this search. Methods are then evauated and their results discussed. Proposal of commonly used enhancements for these methods is made.

Klíčová slova

Vyhledávání podle obsahu, LBP, Lokální binární vzory, HOG, Histogramy orientovaných gradientů, LSH, Hashování podle pozice, min-Hash, barevné histogramy

Keywords

Content-based Search, LBP, Local Binary Patterns, HOG, Histograms of Oriented Gradients, LSH, Locality Sensitive Hashing, min-Hash, Color Histograms

Citace

Martin Kletzander: Vyhledávání opakovaných záběrů, bakalářská práce, Brno, FIT VUT v Brně, 2010

Vyhledávání opakovaných záběrů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Michala Hradiše

.....
Martin Kletzander
17. května 2010

Poděkování

Chtěl bych poděkovat Ing. Michalu Hradišovi za trpělivost, cenné rady a odbornou pomoc při zpracování této práce.

Obsah

Kapitola 1

Úvod

V dnešní době existuje v elektronické podobě mnoho informací a zároveň také několik způsobů jejich vyhledávání. K dispozici máme například fulltextové vyhledávání, vyhledávání na dotazy v přirozeném jazyce a další. Obecné textové informace nejsou jedinými předměty vyhledávání. Vyhledávat lze zdrojové kódy, zboží k prodeji, zprávy a v neposlední řadě i multimédia[1]. Vyhledávání informací z konkrétní kategorie lze provádět na také na specializovaných sítích určených pro tento typ obsahu[2], jako například filmy, obrázky, videa a mnoho jiných. Toto vyhledávání je ale uskutečňováno nejčastěji pouze textem (jméno osoby, název výrobku, název nebo část textu písně). Jinou možností vyhledávání je tzv. vyhledávání podle obsahu, kterému se věnuje tato práce. Zjednodušenou ukázku lze vidět na obrázku 1.1.

Dotaz:	
Výsledek:	    

Tabulka 1.1: Zjednodušený příklad použití vyhledávání

Vyhledávání podle obsahu je v této době na internetu již dostupné[3]. Firma Melodis Corporation, která se věnuje porovnávání a vyhledávání podobných zvuků, spustila projekt Midomi, díky kterému lze na internetových stránkách nebo pomocí aplikace pro mobilní telefon vyhledávat písně nebo zvuky podobné části zvuku nahranému. Dalším příkladem je firma Idée Inc., která provozuje internetové stránky TinEye, kde lze reverzně vyhledávat obrázky. To znamená, že zdrojem vyhledávání nejsou klíčová slova nebo text, ale obrázek, se kterým je porovnávána vizuální podobnost. Firma Google™ spustila projekt "Similar images", pomocí kterého lze vyhledat obrázky podobné jinému obrázku z výsledků vyhledávání. Tyto stránky používají pro nalezení výsledků právě metodu vyhledávání podle obsahu. Pokud by bylo účelem místo obrázků vyhledávat podobné videosekvence, pak jde o porovnávání jednotlivých snímků videa, což jsou opět obrázky, pouze s tím rozdílem, že

[1] vše dostupné na `http://www.google.com`

[2] `http://www.csfd.cz`, `http://www.flickr.com`, `http://www.vimeo.com`, `http://www.youtube.com`

[3] `http://www.midomi.com`, `http://www.tineye.com`, `http://similar-images.google.com`

podobnost musí být stejná u navazujících snímků. Zároveň je při vyhledávání ve videosekvencích nutné uvažovat velké množství vstupních dat (snímků). Vyhledávání a porovnávání takového multimediálního obsahu lze využít například pro automatizaci hledání porušování autorských práv.

Pro vyhledávání podobných obrázků je potřeba definovat jejich podobnost. Z hlediska algoritmizace je hledání podobnosti obrázků ztíženo jejich předchozí úpravou. Obrázky mohou mít rozdílnou velikost, rotaci, jeden může být výřezem druhého, nebo může být méně kvalitní a mnoho dalších faktorů. Druhým problémem ve vyhledávání je velikost prohledávaných dat (jak velikost jednotlivých obrázků, tak jejich množství).

Metody použité pro omezení nároků na paměť a rychlost jsou popsány v kapitole 2, která je rozdělena na část věnovanou popisu algoritmů pro porovnávání podobností (2.1) a část pro zrychlení vyhledávání (2.2). V kapitole 3 je popsán návrh aplikací pro vyhledávání a použité metody. Kapitola 4 se zabývá implementací algoritmů použitých pro tuto práci, popisuje jejich vlastnosti a použité knihovny. Výsledky a efektivita programu pro vyhledávání je diskutována v kapitole 5, stejně jako data na kterých byly programy testovány.

Kapitola 2

Vyhledávání podle obsahu

K úspěšnému vyhledávání v datech je potřeba několika informací. Objekty mezi kterými vyhledáváme je nutné unifikovaným způsobem popsat. Mezi těmito popisy je poté definována podobnost a zároveň také hranice, která určuje zda je porovnávaný objekt ještě podobný či zda už má být považován za rozdílný. Vzhledem k velikosti prohledávané databáze je vhodné na hledání použít jiný než naivní (sekvenční) algoritmus.

2.1 Porovnávání podobnosti

Obrázky lze popsat několika způsoby. Každý obrázek má v sobě uchovány barvy jednotlivých pixelů, které lze bez jakékoliv změny využít při jeho popisu. Pro lepší výsledky je možné převést barvy do jiného barevného modelu (např. pouze do odstínů šedi pokud nezáleží na barvě a podobně). Z obrázku je možné vypočítat po každý bod jeho texturní příznaky, na což je používána například metoda *Local Binary Patterns* (dále jen LBP). Další možností pro reprezentaci pixelu je směr a velikost gradientu v jeho bodě tak, jak je to u metody *Histograms of Oriented Gradients* (dále jen HOG). Z těchto popisů se vypočítají normalizované histogramy a reprezentují tak obrázek jako celek. Tyto popisy jsou takzvanými globálními deskriptory.

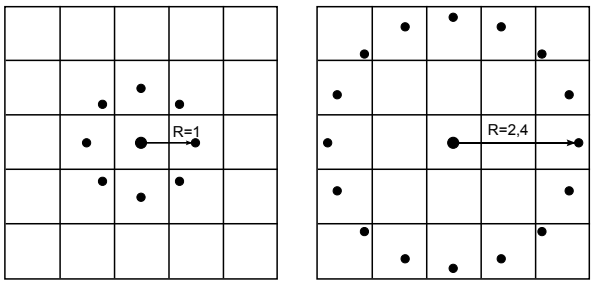
Obrázek lze také popsat tzv. lokálními deskriptory, např. pomocí metody *Scale-invariant feature transform* (dále jen SIFT). Tyto deskriptory označují významné oblasti v obrázku a jsou tudíž invariantní vůči změně pozice v obrázku či jeho natočení apod.

2.1.1 Globální deskriptory

Metodou pro unifikovaný popis obrázku je tedy nejčastěji normalizovaný histogram o předem daném rozsahu, kde jsou pixely reprezentovány buď jejich hodnotou barvy v určitém barevném prostoru nebo pomocí metody LBP.

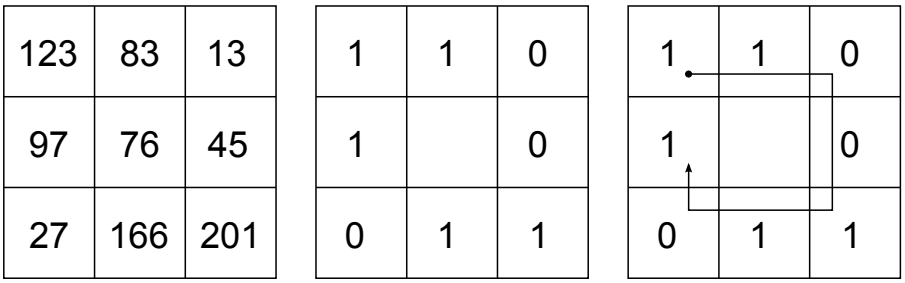
Metoda LBP reprezentuje pixel podle jeho okolí. Vstupními parametry jsou R – poloměr kružnice okolí reprezentujícího pixel a P – počet bodů na této kružnici ze kterých se vypočítá hodnota pixelu. Různé vstupní parametry jsou znázorněny na obrázku 2.1. Pro jednoduchost zobrazení bude dále ilustrována metoda LBP s parametry $P = 8$ a $R = 1$. Dále je vypočtena intenzita v referenčním bodu a v bodech z kružnice (obrázek 2.2(a)). Pokud je intenzita v bodu na kružnici větší nebo rovna intenzitě v bodu referenčním, je tento bod na kružnici reprezentován bitem 1, jinak bitem 0 (obrázek 2.2(b)). Z těchto bitů je nakonec kruhově poskládána hodnota referenčního bodu, jak je vidět na obrázku 2.2(c).

Často používanými variantami metody LBP jsou případy, kdy je okolí považováno za rotačně invariatní nebo se počítá jen s okolím osahujícím dvě bitové změny či kombinace těchto dvou.



(a) Okolí pro $P = 8, R = 1$ (b) Okolí pro $P = 16, R = 2.4$

Obrázek 2.1: Znázornění okolí pro různé vstupní parametry metody LBP



(a) Intenzita pixelů (b) Relativní intenzita (c) Hodnota pixelu

Obrázek 2.2: Binární reprezentace pixelu při použití metody LBP

V [3] použili pro reprezentaci pixelů oponentní model ([7]), sestávající ze tří kanálů – intenzity (I) a dvou barevných složek (O_1 a O_2).

$$\begin{aligned} I &= \frac{R+G+B}{3} \\ O_1 &= \frac{R+G-2B}{4} + 0.5 \\ O_2 &= \frac{R-2G+B}{4} + 0.5 \end{aligned}$$

Tento barevný model umožňuje definování vah zvlášť pro histogramy intenzity a histogramy barevných složek. To znamená, že výsledná podobnost bude jinak závislá na rozdílu barev pixelů a jinak na jejich intenzitě. Dále umožnuje také rychlé a jednoduché přepočítání z modelu RGB.

Metoda HOG ([5]) spočívá v počítání směrů a velikostí gradientů (největších přírůstků intenzity) v jednotlivých bodech obrázku, ze kterých je poté vytvořen histogram. Pro přesnější výsledky je vhodné normalizovat kontrast obrázku. Pro detekci objektů nebo dosáhnutí

větší informativní hodnoty histogramů je obrázek rozdělen na několik částí. Histogramy jsou poté počítány pro každou z těchto částí zvlášť.

Pro definici podobností histogramů lze považovat histogram s rozsahem n hodnot jako bod v n-rozměrném euklidovském prostoru E_n. Jejich podobnost je převrácená hodnota euklidovské vzdálenosti mezi těmito dvěma body. Pro n hodnot histogramů je podobnost P_1 mezi histogramy x a y (kde x_i je i-tá hodnota histogramu x a podobně pro y) definována takto:

$$P_1 = \frac{1}{\sqrt{\sum_{i=0}^{n} (x_i - y_i)^2}}$$

2.1.2 Lokální deskriptory

Metoda SIFT ([8]) popisuje obrázek množinou lokálních deskriptorů. Nejjednoduší podobnost dvou popisů obrázků je poté podobnost přímo úměrná množství stejných deskriptorů a nepřímo úměrná množství deskriptorů rozdílných. Tuto podobnost P_2 lze vyjádřit např. jako

$$P_2 = \frac{|N_x \cap N_y|}{|N_x \setminus N_y|},$$

kde N_x je množina deskriptorů obrázku x a obdobně N_y množina deskriptorů obrázku y. Vzorec lze modifikovat podle toho, jakou podobnost u obrázků vyžadujeme. To znamená, že např. pokud je hodnota podobnosti používána v aplikaci, je problémem rozsah hodnot (pro předchozí vzorec je to interval $\langle 0, \infty)$). Definujeme tedy podobnost P_3 jako nepřímo úměrnou celkovému počtu unikátních deskriptorů (výsledkem je číslo v intervalu $\langle 0, 1\rangle$) a vzorec vypadá takto:

$$P_3 = \frac{|N_x \cap N_y|}{|N_x \cup N_y|}$$

Dalšími metodami může být například ohodnocení vah jednotlivých deskriptorů nebo porovnávání podobností různých podmnožin. Využití vah deskriptorů bylo ukázáno v [2] a [4]. Pokud je $d_w \geq 0$ váha slova X_w, vážená podobnost P_4 je poté:

$$P_4 = \frac{\sum_{X_w \in N_x \cap N_y} d_w}{\sum_{X_w \in N_x \cup N_y} d_w}$$

2.2 Zrychlení vyhledávání

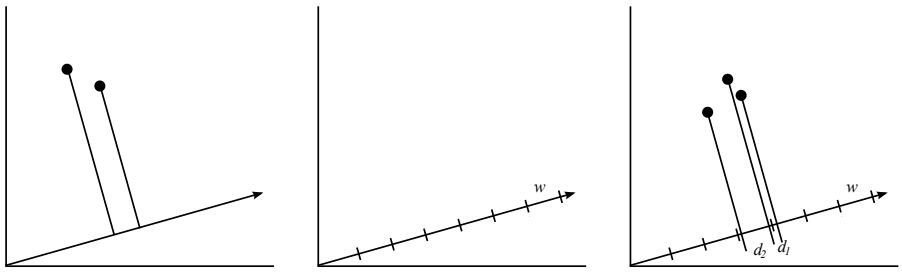
Dalším požadavkem hledání je nalezení výsledků pro dotazovaná data v co nejkratším čase. Sekvenční prohledávání je pro tyto účely pomalé a je potřeba algoritmu pro zrychlení hledání. U popisů obrázků reprezentovaných bodem v n-rozměrném prostoru lze pro zrychlení hledání použít například klastrovací metody nebo hashování podle pozice. Jedním z problémů metod klastrování je heurističnost těchto algoritmů, kvůli které nejsou zaručeny optimální výsledky (velice závislé na počátečních parametrech). Pro tyto účely bylo zvoleno hashování podle pozice.

Pro rychlé vyhledávání v množině lokálních deskriptorů lze použít například metodu *min-Hash* ([1]) nebo její vylepšenou verzi *Geometric min-Hash* ([2]).

2.2.1 Locality Sensitive Hashing

Locality-Sensitive Hashing ([6], dále jen LSH) je metoda reprezentující bod v prostoru pomocí hashe. Tyto hashe se vytvářejí projekcí bodu v n-rozměrném prostoru na vektor (do jednorozměrného prostoru). Znázornění pro dvojrozměrný prostor lze vidět na obrázku 2.3(a). Projekce na tomto vektoru jsou poté rozděleny do částí ("binů") o stejné velikosti w (obrázek 2.3(b). Na obrázku 2.3(c) je znázorněn případ, kdy projekce, které jsou od sebe vzdáleny d_1 (mnohem méně než jaká je hodnota w) spadají do různých binů, zatímco projekce vzdálené d_2 (větší než je d_1, přibližující se w) náleží do binu stejného. Pro omezení pravidelnosti tohoto efektu (při větším počtu náhodných projekcí) lze hranice jednotlivých binů posunout pro každou projekci o náhodnou hodnotu [1]. Výše zmíněná hash je rozumněna "číslem binu" do kterého projekce náleží. Projekcí bodu v na vektor a s veškerými korekcemi (posun o b a rozdělení do binů o šířce w) je hash $h_{\overline{a},b}(\overline{v})$, kterou lze spočítat takto:

$$h_{\overline{a},b}(\overline{v}) = \left\lfloor \frac{\overline{a} \cdot \overline{v} + b}{w} \right\rfloor$$



(a) Projekce bodu na jednorozměrný vektor (b) Rozdělení binů vektoru (c) Nechtěné chování

Obrázek 2.3: Chování projekce bodu na vektor

Body, které jsou blízko sebe, budou mít hash stejnou s pravděpodobností P_s a rozdílnou s pravděpodobností P_d, přičemž platí, že $P_s \gg P_d$. Problémem je, že pro některé projekce[2] je možné, že body, které jsou blízko sebe mohou mít výslednou hash rozdílnou (pravděpodobnost P_d). Pro omezení tohoto efektu je vypočteno více hashí a tyto jsou poté kombinovány – dva body jsou považovány za blízké pokud mají stejnou výslednou hash pro několik projekcí. Například v [9] bylo vytvořeno 150 projekcí a pro každou hash tabulku bylo zkombinováno 7 – 14 z nich. Lze tedy vytvořit hashovací tabulku, kde každý záznam (indexovaný spojením několika hashí) bude obsahovat podobné obrázky. Vzhledem ke konstantní časové složitosti vyhledávání v hashovacích tabulkách zajišuje tato metoda rychlé hledání.

[1] je nutné podotknout, že tímto není efekt eliminován, ale pouze omezen z dlouhodobého pohledu (počítání více projekcí s náhodnými hodnotami)

[2] největší šance nastává pro projekce na vektor vodorovný se spojnicí těchto dvou bodů – největší rozdíl hodnot projekcí

2.2.2 min-Hash

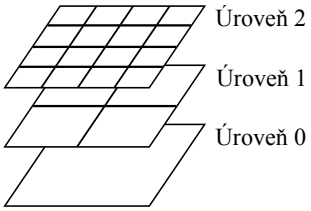
Pokud je obrázek popsán množinou globálních deskriptorů (jako v případě metody SIFT), je možnost použít variantu podobnou LSH, *min-Hash*. Tato metoda bývá používána na hashování podobnosti textových dokumentů ([1]). Prvně je vygenerována permutace π veškerých slov v porovnávaných dokumentech (slovník S) a pořadí slova v této permutaci udává jeho hodnotu. Ze slov obsažených v dokumentu A je poté vybráno to, které má pro danou permutaci nejmenší hodnotu. Tato hodnota je nazývána min-Hash a je definována jako $\min \pi(V_A)$, kde S_A jsou slova ze slovníku S obsažená v dokumentu A. Pravděpodobnost, že dokumenty A a B budou mít stejnou min-Hash je stejná jako podobnost těchto dokumentů podle podkapitoly 2.1. Pro získání kvalitních výsledků nelze uvažovat porovnávání jedné min-Hashe. Náhodných permutací π (a spočítaných min-Hashí) je vygenerováno n a podobnost dvou dokumentů je poté $\frac{s}{n}$, kde s je počet shodných min-Hashí. Zrychlení vyhledávání je dosáhnuto použitím náhodných n-tic min-Hashí jako indexů hash tabulek.

Aby bylo metodu min-Hash možné použít na porovnávání podobnosti obrázků podle extrahovaných lokálních deskriptorů, bylo nutné definovat u obrázku "slovo". V [3] uvažovali "slovo" jako množinu lokálních deskriptorů obrázku, přičemž tyto množiny byly vytvořeny pomocí metody K-means.

Kapitola 3

Návrh

Za účelem demonstrace metod používaných na vyhledávání podle obsahu byly navrženy dvě aplikace. Tyto aplikace by měly být multiplatformní (alespoň Windows + Linux) a mohou využívat doplňkových knihoven.



Obrázek 3.1: Úrovně rozdělení obrázku

Metody použité na vyhledávání jsou založeny na článku [3]. Obrázek je rozdělen na části do tří úrovní podle obrázku 3.1. Z každé z těchto částí je poté vypočten barevný histogram. Tímto je popis přesnější (větší informativní hodnota) a ztrácí invarianci například vůči rotaci a posuvu objektů. Barvy jsou vyjádřeny oponentním barevným modelem s rozdílnými počty binů pro jednotlivé barvy a úrovně (naznačeno v tabulce 3.1). Díky tomuto rozdělení je možné určit prioritu jednotlivých barevných kanálů. Jak je naznačeno na uvedeném obrázku, pro intenzitu je použit dvojnásobek binů, váha tohoto barevného kanálu je tedy dvojnásobná. Veškeré histogramy jsou následně spojeny za sebe a tvoří onen popis obrázku.

	I	O_1	O_2
Úroveň 2	4	2	2
Úroveň 1	16	8	8
Úroveň 0	64	32	32

Tabulka 3.1: Počet binů pro histogramy částí obrázku a barevných kanálů

Pro zrychlení vyhledávání je použita metoda LSH se vstupními hodnotamy založenými také na článku [3]. Pro vytvoření hashí je generováno 55 projekcí a ty jsou kombinovány do 36 hashovacích tabulek. Velikost jednotlivých binů u projekcí je, stejně jako počet projekcí kombinovaných na jednu hashovací tabulku, proměnlivá v obou následujících programech.

První aplikace by měla uživateli graficky přiblížit funkčnost jednotlivých algoritmů interaktivním zobrazením výsledků. Podporované vyhledávání je jak s využitím LSH, tak i

bez této metody. Parametry LSH by měly být nastavitelné za běhu aplikace.

Druhá aplikace by měla automaticky analyzovat efektivitu vyhledávání. První funkcí aplikace je analýza výsledků pro vyhledávání s variabilními hodnotami nastavení metody LSH. Vstupem aplikace je množství nastavení pro tuto metodu a testovací data. Pro poměrnou část testovacích dat se provede hledání podobných obrázků a za správný výsledek je považován obrázek s vzdáleností menší než 200[1]. Výstupem jsou data v textovém formátu, kde pro každou kombinaci vstupních parametrů jsou na řádek vypsány spolu s těmito parametry také statistické výsledky. Jednotlivá pole budou oddělena mezerami.

Druhou funkcí aplikace je analýza efektivity definované podobnosti (vzdálenost histogramů popsaná v 2.1). Tato analýza je provedena procházením záběrů ve videosekvenci. Vstupními daty jsou videosoubor a soubor s anotacemi záběrů v XML formátu. Pro každý snímek z videosekvence je spočítána podobnost všech snímků, přičemž za správné výsledky jsou považovány ty, které náleží do stejného záběru. Z těchto informací je poté vytvořen graf ukazující kvalitu definované podobnosti.

[1] tato hodnota je založená na informacích v článku [3]

Kapitola 4

Implementace

K vývoji programů pro tuto bakalářskou práci byly použity knihovny pro práci s grafickým uživatelským rozhraním, videosoubory, obrázky, statistikami a program na správu verzí zdrojových kódů.

4.1 OpenCV

OpenCV[1] je zkratka pro Opensource Computer Vision a je to knihovna napsaná v jazycích C a C++[2]. Jako projekt začala být v roce 1999 vyvíjena firmou Intel a v době psaní této práce je již dostupná verze 2.1, nyní pod firmou Willow Garage[3]. Jejími výhodami je především dostupnost pro více platforem (Windows, Linux, Mac OS) a důraz vývojářů na její efektivnost a jednoduchost použití v aplikacích. Mezi možnosti jejího využití patří například různorodá práce s obrázky, jednoduchý přímý přístup ke snímkům z videokamer, kalibraci videokamer, jednoduché grafické rozhraní a mnoho dalšího.

V programech k této práci je knihovna používána především na načítání obrázků, jejich rozdělování, převádění mezi barevnými prostory, počítání histogramů a také pro načítání snímků videa ze souboru.

4.2 Qt

Qt (vyslovováno [kju:t] z anglického "cute") je knihovna zjednodušující psaní složitějších a přenositelných kódů. Knihovna je napsána v jazyce C++ a je vyvíjena firmou Nokia (dříve firmou Trolltech). Hlavním účelem bylo vyvíjení grafického uživatelského rozhraní. Jejími dalšími součástmi jsou třídy nahrazující C++ STL datové typy a funkce, se kterými jsou částečně kompatibilní, třídy a funkce pro práci s SQL, XML, vlákny, sítí, multimédii a mnoho dalšího.

V této práci bylo Qt použito zpočátku na grafické uživatelské rozhraní, později na veškeré uchovávání dat a převážnou část práce s nimi.

[1] http://opencv.willowgarage.com

[2] objektově v jazyce C++ je knihovna dostupná až od verze 2.0

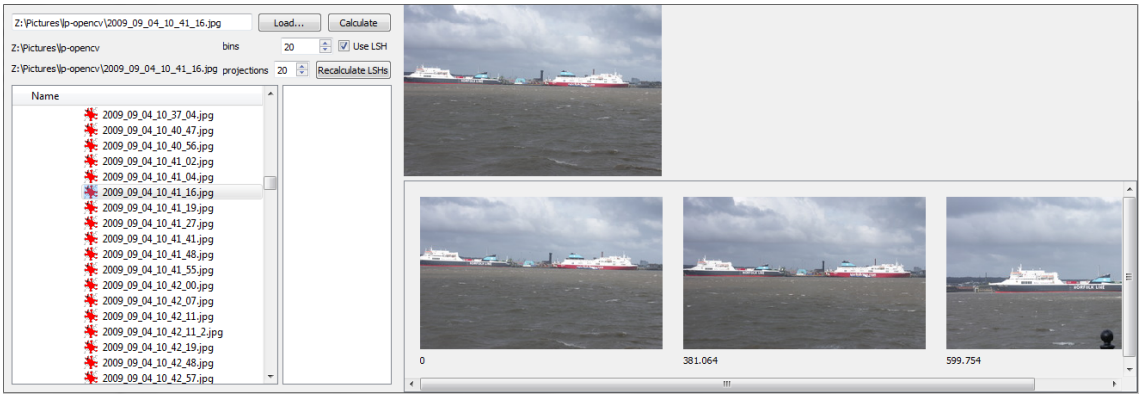
[3] http://www.willowgarage.com

4.3 Ostatní nástroje

Mezi dalšími nástroji, které pomohly při vývoji pomocných programů byla knihovna `TStatistics` od Ing. Michala Hradiše, která zpracovává výsledky vyhodnocování účinnosti algoritmů a pomocí aplikace gnuplot[4] vykresluje ROC (Receiver Operating Characteristic) a PRC (Precision-Recall Curve) křivky. Program gnuplot byl také použit samostatně pro vykreslování grafů v kapitole 5. Na správu verzí zdrojových kódů byl použit program Git[5], který napomohl řádnému udržování změn v průběhu vývoje.

4.4 Program pro testování algoritmů

Pro počáteční testování algoritmů byla vytvořena aplikace, která umí vyhledávat obrázky podle podobnosti. Vzhled aplikace je vidět na obrázku 4.1. V levé části je adresářová struktura, ve které lze dvojklikem vybrat složku a z ní poté tlačítkem `Load` načíst obrázky. Stejným postupem lze načíst první dotazovaný obrázek (aplikace rozezná zda jde o soubor či složku). Tlačítkem `Calculate` lze následovně spočítat výsledky hledání seřazené podle vzdálenosti mezi obrázky. Pod tlačítky `Load` a `Calculate` se nachází nastavení metody LSH. Toto nastavení lze zapnout, což zapříčiní, že po stlačení tlačítka `Calculate` se zobrazí pouze výsledky vyhledané touto metodou, a nebo vypnout, kdy se zobrazují obrázky s nejmenší vzdáleností k vybranému. Počet výsledků je omezen na 100. Zdrojový obrázek a výsledky vyhledávání lze vidět v pravé části okna.



Obrázek 4.1: Rozhraní aplikace pro testování algoritmů

Po vyhledání výsledků umožňuje aplikace kliknout ja jakýkoliv z vyhledaných obrázků, čímž se nastaví jako zdrojový. Opětovným použitím tlačítka `Calculate` jsou vyhledány obrázky s tímto novým zadáním. Při změně parametrů LSH je nutné přepočítat hashe tlačítkem `Recalculate`, čímž se vypočtou nové hodnoty pro načtené obrázky. Výsledky při použití této metody lze opět vyhledat tlačítkem `Calculate`.

[4] http://gnuplot.sourceforge.net
[5] http://git-scm.com

4.5 Program pro analýzu výsledků

Za účelem analýzy výsledků vyhledávání pomocí implementovaných algoritmů byla vytvořena aplikace "analyzer". Tato aplikace běží ve dvou rozdílných módech. V prvním módu aplikace testuje relativní kvalitu vyhledávání pomocí metody LSH při různém nastavení vstupních parametrů. Druhý (anotační) mód slouží k porovnávání správnosti vyhledávání snímků videa ze stejného záběru.

4.5.1 První mód

V tomto módu aplikace projde zadaná umístění, ze kterých načte jednotlivé objekty (volitelně obrázky ze složek či snímky z videa) a do paměti spočítá jejich histogramy. Pro každou kombinaci zadaných parametrů poté spočítá hashe z těchto histogramů a provede vyhledání podobných obrázků pro náhodně vybraných 12.5 % ($\frac{1}{8}$) z celkového počtu načtených obrázků. Aplikace dále zhodnotí výsledky podle počtu korektně vyhledaných obrázků (ty které mají podobnost menší nežli je určená hranice), a obrázků vyhledaných špatně.

Výsledky tohoto vyhledávání jsou poté uloženy do souboru vhodného pro zpracování programem gnuplot. Díky výsledkům pro jednotlivé vstupní parametry lze ve vykresleném grafu pozorovat přirůstající či klesající přesnost vyhledání.

4.5.2 Druhý (anotační) mód

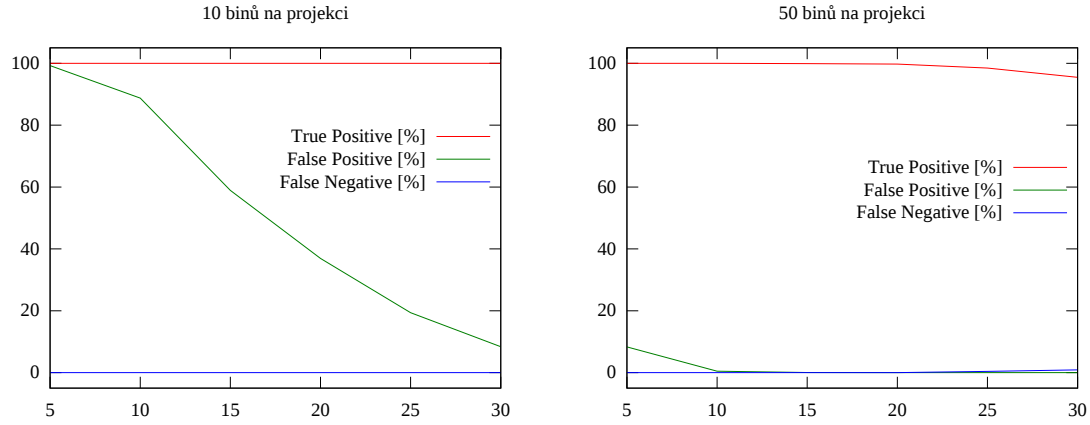
Aplikace v módu anotace přijímá jako parametr jeden video soubor a soubor s anotacemi jednotlivých záběrů v tomto souboru. Pro každý snímek spočítá podobnost veškerých ostatních snímků v celém videu. Podobnost jednotlivých snímků a jejich náležitost do stejného záběru jsou uchovány a pomocí třídy `TStatistics` poté vykresleny do ROC a PRC křivek. Po skončení analýzy vypíše program průměrnou přesnost porovnávacího algoritmu.

Pro porovnání přesnosti vybrané metody byla do aplikace přidána možnost počítání podobností histogramů z LBP. Nejjednodušší metodu LBP se ale nepodařilo úspěšně implementovat jako funkční. Toto bylo pravděpodobně způsobeno šumem v testovacích videosekvencích, což ale nebylo z důvodu nedostatku času dokázáno.

Kapitola 5

Experimenty

Program pro analýzu výsledků byl spuštěn s několik testovacími daty. První sadou dat bylo přes 180 tisíc rozličných obrázků na kterých byla spočítána efektivita metody LSH s variabilními vstupními daty. Za podobné obrázky byly v tomto případě považovány obrázky se vzdáleností menší než 200. Na obrázku 5.1(a) je znázorněno, jak zvýšení počtu projekcí zkombinovaných do jedné hashe snižuje počet vyhledaných výsledků, které nejsou podobné obrázku zdrojovému (False positives). Obrázek 5.1(b) na druhou stranu ukazuje, že vysoké hodnoty tohoto vstupního parametru znehodnocují výsledek při větším počtu binů na projekci. Na obrázku lze vidět, že při nastavení 50 binů na projekci již ve vyhledaných obrázcích není znatelné procento očekávaných výsledků.



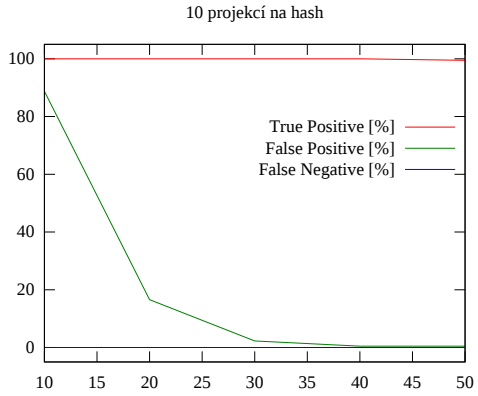
(a) Chování s nastavením 10 binů na projekci

(b) Chování s nastavením 50 binů na projekci

Obrázek 5.1: Chování algoritmu LSH při změnách nastavení počtu binů na projekci s nastavením 10 projekcí na hash

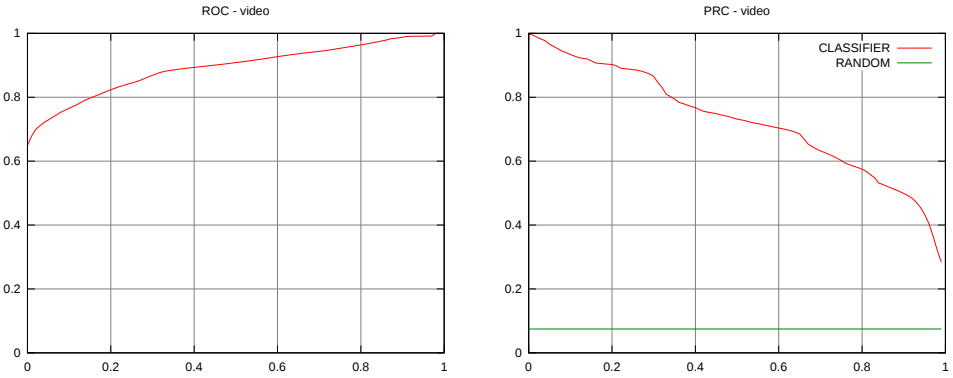
Jak je vidět na obrázku 2.3, snižování hodnoty druhého vstupního parametru metody LSH (počtu projekcí na jednu hash) také snižuje procento false positive výsledků a to dokonce výrazněji. O to dříve ale také znehodnocuje výsledky počtem obrázků nevyhledaných.

Jak již bylo zmíněno v kapitole 4, na znázornění kvality definované podobnosti bylo využito ROC a PRC křivek. Jako vstupní data pro tato měření posloužily mimo jiné záběry z televizního vysílání. Na obrázku 5.3 jsou vidět křivky generované z výsledků pro jeden



Obrázek 5.2: Chování algoritmu LSH při změnách počtu projekcí na hash

z videosouborů. Z ROC křivky lze vyčíst, že s vybranou definicí podobnosti lze dosáhnout výsledků s procentem správně určených snímků nad hranicí 80% přičemž snímků označených jako false positive je pod 20%.



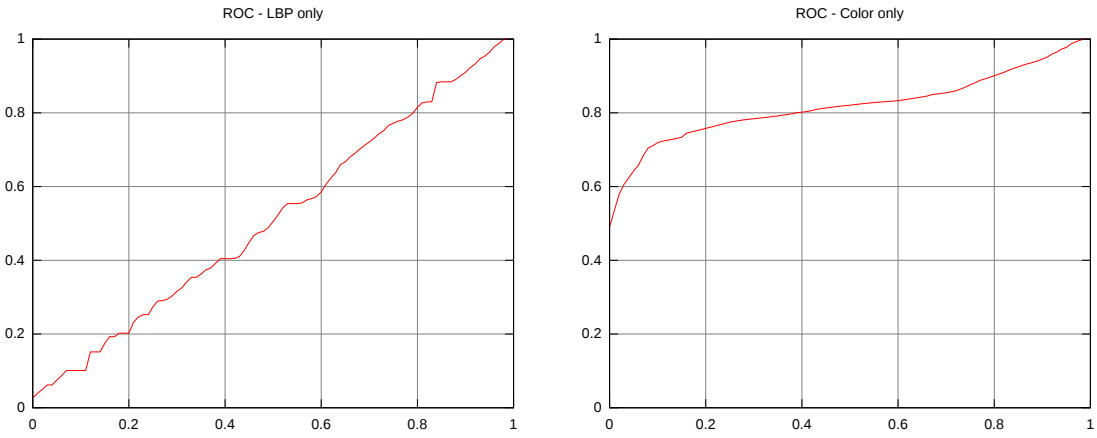
(a) ROC křivka

(b) PRC křivka

Obrázek 5.3: Efektivita algoritmu při prohledávání záběrů

V programu je také implementována podpora LBP. Vzdálenosti histogramů z těchto texturních příznaků ale bohužel nekorespondovaly s podobností obrázku. S přihlédnutím k tomu, že tato metoda se používá na získávání texturních příznaků, je velmi pravděpodobné, že tato chyba byla způsobena obsahem šumu ve snímcích videa.

Šum ve videosekvencích zapřičiňuje změnu intenzity náhodných pixelů v celém obrázku. Na tuto skutečnost je metoda LBP náchylná natolik, že mohla znehodnotit vypočítané výsledky. Pro ukázku je na obrázku 5.4(a) zobrazena ROC křivka vyhodnocení výsledků za použití pouze metody LBP. Pro srovnání je pro stejný videosoubor vidět ROC křivka vygenerovaná za použití metody barevných histogramů na obrázku 5.4(b)



(a) Využití LBP

(b) Využití barevných kanálů

Obrázek 5.4: ROC křivky porovnávající efektivity počítání s barevnými kanály a LBP

Kapitola 6

Závěr

V této bakalářské práci bylo realizováno vyhledávání podle obsahu. Vyhledávání bylo úspěšně aplikované na obrázky i videa. Jedním z vylepšení do budoucna by mohlo být porovnávání a vyhledávání podle lokálních deskriptorů (SIFT, a tak dále) nebo podle jiných metod využívajících deskriptorů globálních (HOG a podobně). Další možností vylepšení je například využití rotačně invariantních LBP pouze s dvěma binárními přechody a porovnání této metody se standardní metodou LBP.

Dále bylo realizováno hledání za pomocí hashovacích tabulek, které disponuje konstantní časovou složitostí a je možné ho tak použít i pro nadměrně obsáhlá data. Vstupní parametry byly diskutovány za účelem upřesnění jejich vlivu na výsledky hledání, avšak optimální parametry nebyly v práci zmíněny. Optimální parametry by měly být zvoleny závisle na parametrech ostatních používaných metod. Zvoleny by měli být například podle velikosti histogramů, velikosti generovaných vektorů, očekávaných výsledků, atd. Do budoucna by mohlo být použito vyhledávání s konstantní časovou složitostí i na ostatní metody porovnávání obrázků.

V praxi by se tato práce dala využít jako základ pro automatizaci detekce porušování autorských práv ve videosouborech. Zajímavým využitím této práce by bylo rozšíření metod a jejich implementace ve větším měřítku. Možnou implementací by mohl být veřejně dostupný projekt demonstrující použité metody veřejnosti (na internetu). Tento projekt by se mohl objevit například jako zadání diplomové práce.

Práce byla velkým přínosem jak obdrženými informacemi tak i získáním zkušeností při práci nejen s pokročilými algoritmy.

Literatura

[1] Broder, A.: On the Resemblance and Containment of Documents. In *SEQUENCES '97: Proceedings of the Compression and Complexity of Sequences 1997*, Washington, DC, USA: IEEE Computer Society, 1997, str. 21.

[2] Chum, O.; Perďoch, M.; Matas, J.: Geometric min-Hashing: Finding a (Thick) Needle in a Haystack. In *IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, jun 2009, s. 17–24, iSBN 978-1-4244-3992-8.

[3] Chum, O.; Philbin, J.; Isard, M.; aj.: Scalable near identical image and shot detection. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, New York, NY, USA: ACM, 2007, s. 549–556, iSBN 978-1-59593-733-9.

[4] Chum, O.; Philbin, J.; Zisserman, A.: Near Duplicate Image Detection: min-Hash and tf-idf Weighting. In *Proceedings of the 19th British Machine Vision Conference, 2008*, London, UK: BMVA, 2008, s. 493–502, iSBN 978-1-901725-36-0.

[5] Dalal, N.; Triggs, B.: Histograms of oriented gradients for human detection. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, Washington, DC, USA: IEEE Computer Society, 2005, s. 886–893, iSBN 0-7695-2372-2.

[6] Datar, M.; Immorlica, N.; Indyk, P.; aj.: Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, New York, NY, USA: ACM, 2004, s. 253–262, iSBN 1-58113-885-7.

[7] Geusebroek, J.; van den Boomgaard, R.; Smeulders, A.; aj.: Color Invariance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník 23, dec 2001: s. 1338–1350, iSSN 0162-8828.

[8] Lowe, D. G.: Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision*, 1999, s. 1150—-1157.

[9] Slaney, M.; Casey, M.: Locality-Sensitive Hashing for Finding Nearest Neighbors. *IEEE Signal Processing Magazine*, mar 2008: s. 128–131.